REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 20, 1995	Final: 9/15/93 - 9/14/94

4. TITLE AND SUBTITLE
Accelerator Systems for Neural Networks, Speech, and Related Applications

5. FUNDING NUMBERS
G: N00014-93-C-0249

6. AUTHOR(S)
Nelson Morgan, Jerome Feldman, John Wawrzynek

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
International Computer Science Institute
1947 Center St., Suite 600
Berkeley, CA 94704-1198

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Dept. of the Navy
Office of Naval Research
Seattle Regional Office
1107 NE 45th St., Ste. 350, Seattle, WA 98105-4631

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
4330/11 ONR 247

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT During the contract period we:

1. Explored the requirements for two new applications in order to improve the capabilities for CNS-1 computation beyond the speech tasks that we have promised for the ONR-URI.
2. Developed library routines to support the general types of computation suggested by the tasks above as well as other tasks we have considered.
3. Further developed object-oriented simulation software to broaden the emphasis of our software support.
4. Studied the design of the subsystems necessary to support high bandwidth I/O on the CNS-1.
5. Simulated the thermal behavior of the largest CNS-1 system we are planning to support.

14. SUBJECT TERMS	15. NUMBER OF PAGES
connectionist computation object-oriented simulation software speech recognition	5
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
			SAR

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

Connectionist Network Supercomputing: Final Report for DARPA funding of expanded CNS-1 functionality

Nelson Morgan and Jerome Feldman
International Computer Science Institute
1947 Center Street
Berkeley CA 94704

John Wawrzynek
U C Berkeley
Berkeley CA 94720

April 18, 1995

0.1 Introduction

This report summarizes the work performed under ARPA/ONR grant N00014-93-C-0249. This grant was part of a larger project, primarily funded by ONR in the form of a URI to Prof. John Wawrzynek at UC Berkeley, to build a connectionist supercomputer called CNS-1. In that project, the emphasis was on system building and on a limited number of applications, primarily in speech and auditory processing. This report is about an extension to the larger project that attempted to make CNS-1 more general and powerful.

During the contract period we:

1. Explored the requirements for two new applications in order to improve the capabilities for CNS-1 computation beyond the speech tasks that we have promised for the ONR-URI.
2. Developed library routines to support the general types of computation suggested by the tasks above as well as other tasks we have considered.
3. Further developed object-oriented simulation software to broaden the emphasis of our software support.
4. Studied the design of the subsystems necessary to support high bandwidth I/O on the CNS-1.
5. Simulated the thermal behavior of the largest CNS-1 system we are planning to support.

0.2 Applications

We have made good progress in exploring additional applications of the Spert/CNS architecture beyond speech processing, working closely with the Arpa programs in Image Understanding. A prototype system for image surveillance applications was executed on the Spert simulator with excellent results.

The work on general purpose parallel programming and the pSather system has gone quite well and the new portable pSather is now being tested.

We also developed routines implementing fixed point FFTs on the Torrent architecture, ranging in length from 32 to 1024 elements. These routines run extremely efficiently - in the computationally intense part of the calculations we achieved an overall average processor utilization of about 75%. This, for example, allows us to compute a 1024 point transformation on real input data in about 12,000 cycles – .36ms with a 33MHz clock.

0.3 Library development

The additional resources made available by the grant allowed a more general emphasis for our library development. In addition to the core low precision fixed point matrix routines used for dense multi-layer perceptrons (which the machine achieves over 500 million arithmetic operations per second on for sufficiently large matrices, and for a 33 MHz clock), we have written the following:

- A set of IEEE standard single precision floating point "millicode" routines. These low level routines can be combined to form a variety of floating point matrix and vector functions. The Torrent fixed point pipe and vector instructions have been leveraged to provide an impressive software floating point performance ($\approx$ 15 MFLOPS for multiplies and adds on a single 33MHz Torrent processor). While roughly a factor of 50 slower than the fixed point operations, these primitives allow the Torrent architecture to be useful in applications where a significant percentage of the computation is not amenable to low precision fixed point calculations. It also permits a more gradual porting approach - a floating point application can run at a reasonable speed with little effort, with only the most time consuming routines converted to fixed point.

- A basic set of integer vector routines. This package includes an efficient integer divide that can execute at a rate of 5 MDivides/S – a speed comparable with the fastest RISC processors available. As with the floating point developments, these both broaden the range of applications that will run efficiently and allow for flexible porting.

- A wider range of function approximation and conversion routines. The emphasis has been on a consistent and generally useful set of functions, rather than just those necessary for our immediate application requirements.

0.4 Object-oriented simulators and software

Our efforts on object-oriented simulation software were marked in the grant period by the relase of version 1.0 of the Sather object-oriented language and its usage by several active user sites. Sather is an object oriented language which aims to be simple, efficient, safe, and non-proprietary. One way of placing it in the "space of languages" is to say that it aims to be as efficient as C, C++, or Fortran, as elegant as and safer than Eiffel or CLU, and support higher-order functions and iteration abstraction as well as Common Lisp, Scheme, or Smalltalk.

Sather has parameterized classes, object-oriented dispatch, statically-checked strong (contravariant) typing, separate implementation and type inheritance,

multiple inheritance, garbage collection, iteration abstraction, higher-order routines and iters, exception handling, assertions, preconditions, postconditions, and class invariants. Sather programs can be compiled into portable C code and can efficiently link with C object files. Sather has a very unrestrictive license which allows its use in proprietary projects but encourages contribution to the public library. The Sather 1.0 system was released in 1994 and has a world-wide user community with an active news group and collaborative projects.

As part of this project, we developed a parallel version to run on CNS and other platforms including networks of workstations. This effort, pSather 1.0, builds on our previous implementation of and experiments with parallel versions of earlier Sather realizations. The goal of pSather is to explicitly support the full range of data structures and algorithms and their encapsulation in object-oriented libraries. During this grant period, we have produced a complete language specification of pSather 1.0 and have incorporated this into the compiler. A preliminary version of the system runs on single and multi-processor Sun workstations. The current effort is focused on porting the full language to the Generic Active Message package. In addition, we are working on several research problems including storage management, object migration and compiler optimizations in the pSather context.

Sather is the basis for our new general purpose connectionist simulator, Icsim and a version of this is running in our lab. Icsim is now being recoded in parallel Sather for use on CNS and other platforms. The Bob simulation package is heavily used and is being ported to the Spert/CNS platform. This will enable users at ICSI and other sites to make an easy transition to Spert/CNS.

0.5 High Bandwidth I/O

A new chip was designed and fabricated to advance our understanding of internode communications issues. Like an earlier chip ("Rambus test chip"), this new one converts a parallel input word to a serial bit stream for transmission at high data rates. A pair of these chips, mounted on a custom circuit board, allows testing innovative circuit design ideas in a system environment.

The new chip ("Rambus II") features full voltage swing signaling, simplifying both the VLSI design and system implementation. Full voltage swing signaling (also called "CMOS compatible") is not popular for commercial chips because of the incompatibility with standard (TTL) levels and the difficulty in designing the driver transistors to best match the expected load.

In our situation, the incompatibility is not an issue, as the chips only communicate with each other. For the driver design, we used an adjustable output driver, a technique used in our first chip, to optimally match the device to the load.

Testing is underway, and early results show the chip attains the target rate of 200 Mbits/second. Several circuit innovations were successful and the design

appears to be robust against extraneous noise.

0.6 Thermal Simulations

Work has continued in the area of thermal management for our chips and systems. The Torrent processors will produce more heat than a comparable size microprocessor because of the nature of the vector architecture. In fact, a primary reason for building application specific VLSI is so that more operations can occur during a clock cycle, but this results in more heat. The safe removal of heat has been the focus of this investigation.

Since the T0 chip has only recently being fabricated, and board tests are still a few weeks away, we are still relying on simulation estimates for our thermal design. Using a range of 8-12 watts for T0, we had a custom heatsink designed and fabricated for use with the Spert (SBUS compatible) boards. To test the operation of the heatsink, a Spert testbed was built using a high power voltage regulator to simulate the heat of the T0 chip, and four sensors to measure the resulting temperature. The sensors monitored the temperatures at the heat source, at one end of the heatsink, and the ambient air in two places.

Results from this testing suggest the custom heatsink will not provide enough heat removal capability for a Spert board mounted in a SparcStation. It appears our estimate for the amount of airflow available in the workstation was optimistic. We subsequently replaced the heatsink with a low profile fan-powered (active) heatsink and ran the tests again. These devices have become quite popular (and inexpensive) recently for cooling overheated microprocessors.

Using the fan-powered heatsink, the temperature target was met for a workstation operating in a lab environment. Which heatsink is used on the final Spert board will depend on the amount of heat actually generated by the T0 chip along with consideration of ambient temperature limits. A simple thermal chamber will be constructed for running tests on the final Spert board at elevated temperatures.

The implications of this testing for the CNS-1 are manifold. Efficient heat removal was a major concern in the original mechanical design of the CNS-1 tower. Maintaining an airflow velocity high enough to keep the Torrent processor cool is easy in the cylindrical tower design, using one large fan as the prime air mover. Conventional electronic packaging, using card racks in a large rectangular chassis, is a poor choice for delivering cool air equally to all the processors in the system. Directing the airflow to the hot processor chips involves intricate ducting and custom heat sinks, and yields unpredictable results.

The use of fan-powered heat sinks changes everything. Although problems remain with building a multiprocessor in conventional card racks (internode cabling and I/O are two important ones), active heat sinks greatly simplify the cooling. Essentially, the local fan provides a heat removal mechanism for the processor chip that is (almost) independent of the packaging method. We hope

to use this information to simplify the final system design of the CNS-1.

0.7 Conclusions

This grant provided funding for useful extensions to the basic CNS-1 project. Perhaps one of the most important of these, in terms of the longer-term viability of the project, was the extension of the underlying software to include more generally applicable routines such as IEEE floating point support and integer vector operations. Additionally, under this grant we learned that we could use the Torrent effectively for a number of tasks that were not explicitly connectionist, such as spectral estimation and disparity calculation for stereopsis. The grant also enabled us to extend and accelerate the work on parallel software that can be used on a variety of platforms in addition to CNS. It also supported an investigation of coding requirements that led to the design of an active heatsink.

Note that all performance figures quoted in this report are conservative, based on a worst-case estimate for a T0 clock at 33 MHz. The actual chip may be significantly faster.

Finally, as of this writing the T0 chip has returned from fabrication and has passed all wafer tests (for a reasonable yield of the dice).